KAUDroid

A tool that will spy on applications and how they spy on their users

Adrian Carlsson, Christian Pedersen, Fredrik Persson & Gustaf Söderlund

Faculty of Health, Science and Technology

Computer Science
KAUDroid

A tool that will spy on applications and how they spy on their users

Adrian Carlsson, Christian Pedersen, Fredrik Persson & Gustaf Söderlund
KAUDroid - A tool that will spy on applications and how they spy on their users

Adrian Carlsson, Christian Pedersen, Fredrik Persson & Gustaf Söderlund

WORKING PAPER | Januari 2018


© The authors

Distribution:
Karlstad University
Faculty of Health, Science and Technology
Department of Mathematics and Computer Science
SE-651 88 Karlstad, Sweden
+46 54 700 10 00

Print: Universitetstryckeriet, Karlstad 2018
KAUDroid

A tool that will spy on applications and how they spy on their users.

Adrian Carlsson
<adriancarlsson@hotmail.com>

Christian Pedersen
<christian.pedersen@telia.com>

Fredrik Persson
<fredpers104@student.kau.se>

Gustaf Söderlund
<gustsode102@student.kau.se>

Master of Science in Computer Engineering

23rd January 2018
Abstract

Our mobile phone is today our most precious and confidential part of us. In many cases we do not only share phone number with it, but also our most private information. This information is to some extension naively shared and gathered without our knowledge. As the number of advanced and sophisticated services increase the more sensitive information we seem to throw at them. To prevent privacy intrusions Googles’s operation system Android uses something called permissions. Permissions are a decision to allow an Android application access to a resource on a specific device. Once a permission is granted there is no possibility for a user to verify that the application does not take advantage of the given trust. With granted permissions applications can collect a vast amount of information without any connection to the service it provides. To highlight this problem we present the KAUDroid platform, a two part, Android application surveillance system. KAUDroid consists of an Android application that collect permission usage on phones and a central server responsible for data storage. Information is presented to the public through a web user interface using graphs to visualize data in an understandable manner. With this tool we hope to raise the general awareness of how third-party applications tend to abuse their trust and to help people recapture their unconsciously lost privacy.

Contents

1 Introduction 1
   1.1 Awareness . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
   1.2 Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
   1.3 KAUDroid . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

2 Background 2
   2.1 Android . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
      2.1.1 Intents . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
      2.1.2 Permissions . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
      2.1.3 Services . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
      2.1.4 Intentservice . . . . . . . . . . . . . . . . . . . . . . . . . . 5
   2.2 Development tools . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
      2.2.1 Android Studio . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
      2.2.2 Eclipse . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
   2.3 Server . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   2.4 SQL server . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   2.5 Graph framework . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   2.6 Programming language . . . . . . . . . . . . . . . . . . . . . . . . . 7
      2.6.1 Java . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
      2.6.2 HTML . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
      2.6.3 JavaScript . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
   2.7 Git . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
   2.8 Spark . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
2.9 Thymeleaf ................................................................. 8
2.10 Maven ................................................................. 9

3 Design ................................................................. 9
3.1 Android Application ................................................. 10
3.1.1 GUI ................................................................. 11
3.2 Database ............................................................. 13
3.3 Server ................................................................. 14
3.3.1 HTTP Server ....................................................... 15
3.3.2 Privacy Server .................................................... 15
3.3.3 Administration tools ............................................. 16
3.4 Web User Interface .................................................. 17
3.4.1 Data selection and preparation for visualization .......... 18
3.4.2 Graphs ............................................................. 19
3.4.3 Pie Chart ......................................................... 19
3.4.4 Stacked Column Graph ........................................ 19
3.5 Discussion and assessment of design .......................... 22

4 Implementation ....................................................... 22
4.1 Android Application ................................................ 22
4.1.1 MainActivity ...................................................... 22
4.1.2 SendDataService ................................................. 23
4.1.3 LocationService .................................................. 23
4.1.4 loggerService ..................................................... 23
4.1.5 deleteSyncedDataService ................................. 24
4.1.6 Cryptography ..................................................... 24
4.2 Database ............................................................. 24
4.3 Server ................................................................. 25
4.3.1 HTTP Server ...................................................... 26
4.3.2 Privacy Server .................................................... 27
4.3.3 Admin tools ....................................................... 29
4.4 Web User Interface .................................................. 29
4.4.1 Graphs ............................................................. 29

5 KAUdroid Project .................................................... 30
5.1 Development of the Project ...................................... 30
5.1.1 Start Up .......................................................... 30
5.1.2 Taking action ..................................................... 33
5.1.3 Mid Project ....................................................... 33
5.1.4 Wrap up & Report .............................................. 34
5.2 Roles ................................................................. 34
5.2.1 Project Manager ............................................... 34
5.2.2 Technical project manager ................................... 34
5.2.3 User Interface designer ....................................... 34
5.2.4 Software tester ............................................... 35
1 Introduction

When you install a new application on your phone it will require a set of permissions. These permissions could be location related, reading data from storage and more. One would assume that these permissions are required for the application to work as intended and only that. According to [1] some applications can use permissions in a questionable manner, intruding on the privacy of the mobile owner. If this is the case, how can it be allowed to happen? There are many articles in the media that discuss how applications require permissions and what that means. But there is not much to read on how the applications behave when installed and how often they use permissions. Could this be what is missing for people to react to it, proof of how the applications behave? How can this be provided to a regular user?

If the user is unaware of the problem nothing will happen. The user need to be able to understand what is happening, before a reaction can be started. To be able to create awareness, we need data about application behavior that can make the user aware of what is happening without their knowledge. This project implements a prototype for data collection about how applications use their permissions.

1.1 Awareness

How is awareness created in the most efficient way? It has to be accessible for people and in a way that is easily understood. Using complex graphs and technical words will not help. Even though the graph might show a lot of data, most people will not spend the time needed to understand it. People should be able to gaze at a graph and understand it as well as identifying potential problems.

This is easier said then done. There is a lot of different data to show and a lot of combinations. To many options will also confuse the user and will work against creating awareness. So, what is needed is a simple service that can gather and show data in an easy and understandable way to the user.

1.2 Data

Collecting the data is the first step, to do this you need a rooted Android mobile device. This is a problem, since most people do not root their device’s. Because of this the data from the applications has to be publicly available so that other people can look at data that is collected by people who have rooted device’s.

---

1 A device with root access, allowing access to the operating system code. Allows for modification of software code.
This data then has to be represented in a fashion that will make it easy for people to understand it. It also has to show information that will point out how an application uses the permissions. That is how often they are used, when they are used, etc. The data has to be simple but still able to satisfy the user’s curiosity as to what happens.

1.3 KAUDroid

KAUDroid is a platform that will gather data of how different applications use permissions. This data is then stored on a remote database. All the data that is collected will then be available for interested parties and visualized with the goal of increased transparency. The goal is to have a tool that can help in creating awareness regarding the issue of mobile applications possibly spying on you.

2 Background

In this chapter short information about some of the different technologies used during the project will be presented. In addition to the technologies used the reason those were chosen will be presented.

2.1 Android

Android has quickly become one of the world’s biggest operating systems (OS) for phones with a market share of above 80% [2]. This popularity together with that iPhone OS (iOS) is much more restricted in what you are allowed to do as a developer. Since there was a prototype available for the project written for Android, this made the decision to keep using Android quite clear.

Android is a Linux operating system where each application is run in a separate sandbox within the Android system to separate them from each other. This is made to improve the security in the OS since the development of new applications is open for everyone and no application should interfere with another. In order to communicate with other applications so-called intents (see section 2.1.1) can be sent through the OS and then the application can answer. But it does not have to. For applications to use system resources the application needs to ask permission for a specified resource before use, see section 2.1.2 [3].

2.1.1 Intents

Intents in Android is the way to send messages between otherwise isolated applications and within an application between activities and services. This makes intents the so-called glue in Android that is used to bind everything together and a frequently used within all applications. The main purpose of the intents is to start other activities and services. The intent’s can contain the action to be performed, recipient and some data. These intents are then sent by the
OS to the desired recipient, if one is not specified it gets broadcasted and the applications that can perform the task answers. An example of that can be if another application want to open a web page The application can send an intent to the OS and those applications that have that functionality answers instead of that the application build its own browser functionality 1,5 see figure 1.

Figure 1: Applications answer to an intent.

2.1.2 Permissions

In Android the security is enforced with the help of runtime permissions. In order for an application to use certain resources it needs to have the right permissions for those resources. For the application to acquire the permissions, it needs to ask the user for them. Up until Android 6.0 (API (application programming interface) 23) this was done only at installation time 6. Now this is achieved with runtime permissions, see figure 2 for an example of a runtime permission popup.
Figure 2: Runtime permission dialog example.

There are currently three different threat levels for permissions in Android [7,8]:

- Normal permissions, which is permissions that dont pose any risk for the user economically or impose on privacy.
- Dangerous permissions is permissions that can affect the user economically and impose on privacy. An example of a permission here is send text messages ($SEND_{SMS}$).
- Signature/system permission used for the system and should not be granted to specific applications, but can be used in rooted phones or with specific certificates. Permissions in this category is for very trustworthy system components only.

2.1.3 Services

A service is one of Androids built-in components that a developer can use to perform work in the background independent from the rest of the application without a user interface (UI). So even if the user switches to another application the service will continue to run. In order to stop a service the application need to call either stopSelf() or stopService() for services of type one and type two see 2.1.3 Which is built in functions in Android.

For services of type 3 they will be destroyed when all components bound to the service are destroyed. The different types of services are [9][10]:

1. Foreground services
   - A service that has a status bar icon. For example a pause/play button for streaming applications

2. Background services
• A service that runs in the background and is not noticeable to the user.

3. Bound services

• A service that is bound to some other component. Can for example be used for communication between an activity and a service. This service is destroyed when all the bound components are destroyed.

2.1.4 Intentservice

Intentservices is a special type of service that is unlike regular services, see section 2.1.3, does its job in a separate thread instead of the main UI thread. Another difference is that Intentservices finishes when the job is done, so there is no need to stop it explicitly [9,11].

2.2 Development tools

To develop and work with the Android application, the server, the database and the web user interface some sort of development tools were required. In this subsection the selection on which development tools were used during the project are described. A summary is made of the functionality and the rationale for different choices are presented in this section.

2.2.1 Android Studio

Android studio is the official integrated development environment (IDE) for the Android OS. It has a wide set of built-in tools the developers can use. Such as the profiler to monitor memory, CPU and network traffic. There is also debugging, instant run, drag and drop layout creator and the possibility to create emulators for a wide range of different phones, screen sizes and android versions [12]. This made the choice to use Android studio quite easy because the start project was written in Java and since we do not need support for multiple OS such as iOS or Windows there is no need to use Visual studio with Xamarin [13]. Eclipse was not chosen since Android studio is the official IDE and the setup is easier since no plug-ins are necessary for Android Studio.

2.2.2 Eclipse

Eclipse is a Java-development platform. With this IDE it is possible to build projects in multiple different languages and not only in Java. Though, it is mostly used for Java applications, the available plug-ins for Eclipse makes it possible to write applications in almost every language. Furthermore, the plug-in mechanism allows Eclipse to be able to work with multiple different kinds of application such as, database management systems, modelling tools and network applications, among others. It has a highly customizable UI, being able to move panels, menus and tool bars as wanted, which make it possible to suit every kind
of developer. It is a well used IDE in the industry and is completely free to use, which of course is a big reason to its success\textsuperscript{14,15}.

Eclipse was chosen during this project mainly because of the fact that the server code would be written in Java. It is the most widely used IDE for Java application development and the team had previous experience with it.

2.3 Server

The server runs Ubuntu 16.04. It is a Linux based OS and the server is accessed remotely and operated through a terminal. Ubuntu had support for running MySQL and for Java programs and all group members had previous experience with Ubuntu. It is also easy to work with and offers great control. This combined with that it is open source made it a good decision.

2.4 SQL server

MySQL is a relational database introduced by Michael Widenius and had its first initial release 23 May 1995. It is a widely used database software and is used by companies like Facebook, Twitter and Youtube to efficiently help them with their work. It is designed and optimized for Web applications, which has lead to its big success.

A database is beneficial to use for millions of different reasons and together with it, a good database management system is needed. As mentioned above, MySQL is a relational database. By being a relational database it prefers to save information in multiple different tables, related to each other, instead of putting everything into one big table. It offers the developers the possibility to set up rules on how the relationships between the tables should be to make it work as the developers want\textsuperscript{16}. It uses the widely used Standard Query Language (SQL) to access databases and may be used either by entering it directly or create SQL statements. It is easy to use, fast, scalable and reliable. It offers a rich set of functions and requires little to no attention. It is multi-threaded, which means it includes the possibility to handle multiple requests simultaneously\textsuperscript{17}.

2.5 Graph framework

To render graphs Highcharts is the selected framework. As mentioned in section 2.6.3 the selected language for rendering of the graph is JavaScript. After that the next step was to select a graph framework that would be able to render graphs on dynamic data. The framework should also be able to render a pie chart and stacked column graph. Different forums were consulted in what frameworks were popular and easy to work with. This is were Highcharts was mentioned, this specific framework could render the graphs dynamically and had a nice aesthetic feel to it. This is the reasons why Highcharts was used to render the graphs. Highcharts also had a good documentation and several different examples of their graphs with code, which made it easier to work with.
2.6 Programming language

To build KAUDroid a selection of programming languages (PL) have been used. Today there are many well developed PLs available for the developers. Each of them has different strengths and weaknesses. Selecting the right PL can make the development of the platform a lot smoother. Below are the reasons why the specific languages were chosen for KAUDroid.

2.6.1 Java

The programming language that was used during the project was Java. Java is a popular programming language in today’s society and it offers great built-in tools. It is an object-oriented, easy to use language which can help you quickly produce code. It introduces a good exception handling which helps a lot when dealing with error conditions which is extremely helpful during development. With a goal of having as few dependencies as possible, it is automatically designed for maximum portability. It is possible to run compiled code as long as it has the Java virtual machine installed \cite{18}.

Both the server code and the application code is written in Java. The reasons behind that is firstly, the application shell that was provided from previous work was written in Java. Because of that it would be simple to continue writing in Java for consistency. It would make it easier to just use the good functionality that was in it from the beginning without having to change that much. The decision of having the server written in Java was because it was preferred in the project description. Most of the project members had previous experience with Java.

2.6.2 HTML

For the website HTML was the selected language. The reason for this is because it is a widely used language on the web. Also this is where the project members had previous experience, which made this the easier choice as well. There were no limitations that HTML put on the project. This resulted in HTML being an optimal choice for this project.

2.6.3 JavaScript

JavaScript was the language used to render the graphs. JavaScript was selected because it allows the client to render the graphs locally, allowing interaction with the graph. It also decreases demand on the server, since the server does not have to render any graphs. Clients today are powerful and should not have a problem to render a simple graph as well. JavaScript was the option that works great for client rendering and it is a widely spread programming language. With previous mentioned arguments this made JavaScript the perfect option.
2.7 Git

Git is a distributed version control system which is widely used nowadays. It was developed 2005 and is a quality open source project. In addition to being distributed, it is also developed with flexibility, performance and security in mind.

A version control system is a system which helps software developers manage changes on their code over time. Developers that are working in a team are continually writing new code or making changes. Some people may work a new feature while other tries to fix a bug in the old source code. Everyone makes changes in different parts of the project and these changes does of course make several files change. A version control system is there to help the developers solve problems that may occur when working concurrent on different parts of a project. A version control software is an essential part of today’s development of modern projects. Developing software without using some kind of version control software is risky, it is like having no backups. When using Git, every developer in the the project can make changes in their own branch. It provides the ability to work on your own branch without having an effect on someone else. When the changes is done, you can simply just merge the changes into a collective branch and verify that the changes made on the different branches do not conflict. Having this overview over every change that has been made on every little file makes it easy to trace and understand what other project members are doing under development [19].

Git was chosen because it accommodates distributed programming and asynchronous development in the group. In addition, the team were already familiar with Git and it was provided by the university. Every project member had their own branch which they could work within, and at the end of the work day, changes that was considered done were merged into a collective branch. It made it clear on what was done during the day.

2.8 Spark

Spark is a Java based web application framework. The framework is open-source and free to use. By default the Spark framework is built on the Jetty web server [20]. Spark is an alternative to Spring MVC and other similar web development frameworks. A lightweight structure and a design for easy development with minimal effort are two properties that validates the decision to use the Spark framework for the KAUDroid web implementation.

2.9 Thymeleaf

Thymeleaf is an engine that is based server-side. It has a lot of functionality, but it is mainly used for two purposes in the the KAUDroid platform. It is used to pass data between the server and client in an efficient way. Thymeleaf can also make the design a lot better and smoother. It can for example render
dynamic tables. In KAUDroid it is also used to hide scripts when they are not used or hide div tags with its content.

2.10 Maven

Maven is a tool that aims to simplify a number of aspects for Java projects. According to [21] the main objectives are:

- "Making the build process easy"
- "Providing a uniform build system"
- "Providing quality project information"
- "Providing guidelines for best practices development"
- "Allowing transparent migration to new features"

Several of these bullet-points are interesting for this project. The first bullet-point, "Making the build process easy" made it a lot easier to constantly build the project on the server. It only requires that Maven is installed on the server. The second bullet-point, "Providing a uniform build system" was a great help when development of the project was done by more than one person. When a new dependency was added it was added to a .pom file which Maven then ran and updated the local project with the new dependency. So, if a new dependency is added it requires only one change in a single file, then all project members could use the new dependency. This helped reduce the complexity of the development process. The other bullet-points had an impact on the project as well, but it was the two first that were the main reason Maven was used.

3 Design

The following section explains the overall infrastructure of the system, how different modules are related to each other and how they communicate. It also explains the tactics used and the design decisions taken during the development process to solve particular problems. Choice of IDEs and programming languages are also justified for the given contexts.

The KAUDroid platform is a multi-part project. The goal of the platform is to create awareness regarding how applications use permissions. It is important that the implementation is done right to maximize effectiveness, improve user awareness. From an abstract point of view the project is split in two major parts. The Android application [3.1] is the information gatherer, it periodically queries the installed applications on the phone where it is installed. Application permission usage is then stored on a local file. The unsynced information in the local file is periodically sent to the server encrypted over Internet Protocol (IP)

Divides a HTML page into different sections where each div tag is logical container. A div tag can contain more div tags to create a subsection.
traffic. The server (3.3) is the most central part of the platform. It is responsible for storing the information sent by clients and to provide a way to present the stored data to users. The server is split into four submodules, for more implementation details of the different parts see their respective subsections. Figure 3 shows an overview of the system infrastructure.

Figure 3: KAUDroid infrastructure overview

### 3.1 Android Application

The application has two main purposes, to gather data about permission usage and to send that data to the server. To do so there are one service to gather data and one intentservice to send it. Besides the applications main purposes it has some other functionality which includes:

- A settings menu to make it more general see figure 5 on setting options.
- Deleting of synced entries to not fill the phones memory.
- Open a browser to show data in graphs.
- Transparency control
- Location blur
- Get a JavaScript Object Notation (JSON) Log
- Remote Privacy.

Of this functionality the application provided by Nurul Momen [1], had the functionality of the last four bullet points above and to gather permission data. The focus was to extend the data gathering to include the global positioning system.
(GPS) coordinates and to send the data gathered by the application automatically encrypted to the server. After that goal was achieved the refinement and settings functionality was added.

3.1.1 GUI

The main focus for the application was the functionality but some changes were made to the GUI from the original application. Two more buttons were added to the home screen, see figure 4. But the main difference is that the layout is now designed to scale up and down depending on the users screen sizes. Instead of the fixed sized layout that was implemented originally.

![Home screen](image)

Figure 4: Home screen

If the settings icon is clicked the user arrives at the screen in figure 5. The layout for the setting activity is simple to navigate and designed with the use of Androids preference API. This makes it easy to add more settings in the future and gives the users a familiar view of how settings are supposed to look, see figure 5 [22].
If the different options in figure 5 are clicked the user is presented the views in figures 6, 7, 8, 9.

**Figure 5: Settings**

**Figure 6: Data & sync clicked**

**Figure 7: Cleared synced logs clicked**
3.2 Database

The users will be sending a lot of data to the server from their phones and the data will be containing different information depending on different users. A database design which can receive and handle large amounts of data simultaneously arriving from multiple clients is then needed. The decision to implement a database at the server was made. A database introduces important capabilities so that data from the users may be stored in an effective way, as well as fetched for analyzing and providing feedback to the users. More specifically, a MySQL database was chosen, with the justification that it is easy to use, well-documented and universally used. Furthermore, the development team is familiar with it from past projects.

The database functionalities are located in its own module, split among three different classes that fulfill different needs. The decision to have it in its own module follows the single responsibility principle and only one of the three classes are accessible by other modules, to keep it as simple and independent as possible.
Figure 10: Illustration of the database design.

The database consists of three tables, which is illustrated in figure 10, together with their relations to each other. The decision to not save every piece of information in the same table is to prevent the case of having a lot of redundant data in one table. To prevent having the same combination of phone, application and permission added in the table again, for every time it is called, the timestamps is instead moved to an external table. Furthermore, the timestamps are being saved and stored as datetime variables. A datetime time variable contains both a date and a time part which is exactly what is wanted. It is interesting to both see the dates as well as the times during the day to be able to see patterns and interesting behaviors.

The decision to save GPS information in an own table is to prevent having null values in the tables. This would be the case if the GPS information was for example saved together with the timestamps. Since not all permissions includes GPS coordinates.

3.3 Server

The monolithic architectural pattern chosen for the server gives both simple development and deployment [23]. Many of the advantages given by an Service Oriented Architecture (SOA) will not be noticed until the application becomes large. With large meaning both in number of code lines and also seen from the number of application users [24]. The easier reuse of program components is not enough to outweigh the added complexity given by a SOA. One major drawback of the monolithic architecture is that it makes continuous deployment more difficult. It can be seen as an obstacle to frequent deployment, this is because the modules in the monolithic architecture has higher coupling.
3.3.1 HTTP Server

The HTTP module fetches information from the database and sends it to the web client for visualization. In order to keep modifiability and maintainability as easy as possible a web framework was used. Criteria for this framework were low complexity and lightweightness. Based on the previous requirements and the development groups experience, a decision to use the Spark framework was taken.

Deciding how to render graphs was another important design decision. The information could either be rendered as a picture on the server and sent as a picture or the data could be sent and used for graph rendering. The drawback of sending data to the client is that this design might not be convenient for large data sets. For information about graph rendering at client side see section 3.4.

3.3.2 Privacy Server

In order to keep the maintainability high during the development of the Privacy server this part of the code was designed to offer modifiability and modularity. This is achieved through careful class abstraction and following the single responsibility principle, that it should be possible to explain the purpose of a class in just one sentence.

To make sure that data had been correctly received by the server an easy plain text protocol was used. The protocol was designed to use two different messages, Ok and Invalid. If the server receives Stop it disconnects the client and sends Ok to the client. If the data received is something else the server checks if the input is valid. Depending on the result the server answers either Ok or Invalid. Figure 11 visualizes the protocol design in a state-diagram.
Another decision made was the introduction of encrypted communication between the server and the Android client. The reason for this was that the task specifica-
tion, see appendix C, stated it. Since the database and the communication between the server and the web client is still in plain text and for compatability a decision was made to only use 128 bit encryption keys.

The system is not designed for an significant amount of users, but when in-
troducing more then one user the decision to use a multi-threaded tactic was
taken. Even if this type of resource management tactic can cause concurrency problem it seemed mandatory with a multi-threaded application for this type of multi-user system.

To be able to disconnect the connected clients from the server in a secure way a decision was made to introduce a list with information about all the clients, both a reference to the running thread and the created object. Each client thread is independent of each other except for reading and writing to the database. The MySQL database, see section 3.2, is multi-threaded which means that no design decisions have been taken in order to synchronize the different threads.

3.3.3 Administration tools

The Administration tools are designed to be easily extendeble and invite to continued development of the system. At first the Administration tools were meant to include functionality for database management. But the flexibility and simple usability of the MySQL command line API showed that mapping this to the KAUDroid system was unnecessary, both from a user and developer perspective. Even if the Administration tools are a small part of the system it
is separated within its own class to increase both modularity and modifiability. By hiding information inside this class the overall maintainability becomes less hard when adding new administration capabilities. Isolating the code and giving the class a single responsibility also enhances the possibility to easy future modification.

3.4 Web User Interface

For a example views of the Web UI see appendix [X].

The web UI is the part of the platform that will visualize the data and make it easy for people to understand it. Graphs are mainly used to visualize the data to the user. There is a set of options that the user can use to filter what data should be visualized. These options are presented in the form of a drop-down menu where the user can select from specific options. There are three different drop-down menus:

- **Device**: select what device you want to see data for, there is an option for all.
- **Application**: select what application you want to see data for, there is an option for all.
- **Permission**: select what permission you want to see data for, there is an option for all.

These drop-down menus can be used to visualize different data sets. Device selection always has to be chosen and to chose permission a selection of applications has to be done as well. This will result in many combinations:

- **All Devices**: Will show how many permissions each application has used for all devices combined.
  - **All Apps**: Will show what different permissions and number of times used for all applications on all devices.
    - **All Permissions**: Will show how many times all permissions has been called on by all applications, over the last 24 hours on all devices.
    - **Specific Permission**: Will show how many times a specific permission has been called on by all applications, over the last 24 hours on all devices.
  - **Specific App**: Will show what different permissions and number of times used for one specific application on all devices.
    - **All Permissions**: Will show how many times all permissions has been called on by a specific application, over 24 hours on all devices.
    - **Specific Permission**: Will show how many times a specific permission has been called on by a specific application, over 24 hours on all devices.
• **Specific Device**: Will show how many permissions each application has used on one device.
  
  – **All Apps**: Will show what different permissions and number of times used for all applications or one device.
    
    * **All Permissions**: Will show how many times all permissions has been called by all applications, over 24 hours on one device.
    
    * **Specific Permission**: Will show how many times a specific permission has been called by all applications, over 24 hours on one device.
  
  – **Specific App**: Will show what different permissions and number of times used for specific application over one device.
    
    * **All Permissions**: Will show how many times all permissions has been called by a specific application, over 24 hours on one device.
    
    * **Specific Permission**: Will show any many times a specific permission has been called by a specific application, over 24 hours on one device.

Adding this many combinations will cause confusion unfortunately. But at the same time they have an importance to them. With all of these combinations the user can start with an abstract view to easily detect a deviation from the expected. But then also zoom in to get further information on what is causing this deviation. But this increased information comes at the price of the increased complexity. There is also an option to select a time frame, this is to enable the user to reduce the set of data. You can only select a time frame for the stacked column graph, for graph description see section 3.4.4.

![Figure 12: The drop-down menus with the time frame input.](image)

### 3.4.1 Data selection and preparation for visualization

As a user sends a request for data the HTTP server [3.3.1] will receive it and it will interpret what data to send. The HTTP server will then pass back the requested information in a HashMap. The HashMap is used so that multiple variables can be passed simultaneously. The graph data is stored in JSON format in the HashMap and is reconstructed to JSON objects with JavaScript. Then the data is used to construct the graphs.
3.4.2 Graphs

As discussed in section 1.1 it is important that the graphs are simple but still illustrate a specific behavior of applications with respect to end-user privacy. There are many methods to display graphs through a browser and it was important that a good method was chosen. First decision is if the graph should be rendered on the server or on the client. The difference is that with client side rendered graph they can be interacted with. If the graph was rendered on the server it would probably be sent as a static object and no interaction would be possible. Also server rendered graphs would increase performance demands on the server. This led to client rendered graphs being the optimal solution.

The next decision is the method of rendering on the client and what language should be used. Here the decision became JavaScript. The reason for this is because JavaScript is available on all modern browser’s, unless it is blocked by the user. JavaScript will not require the user to install any additional software, making it simple and easy to use. To create the graphs a framework called Highcharts was used, which had the support for rendering of dynamic data and the graphs that was sought for.

Data would be passed from the server to the client for the client to render the graph. The format that the data has is JSON, this was selected because Highcharts had great support for working with JSON data. The JSON data is then parsed properly and rebuilt into a format that the Highcharts framework can work with. An example of a graph can be seen in figure 13 and figure 14 below. Each pie slice can be highlighted on the web page to see detailed information.

Two different types of graphs are used, pie charts which can be seen in figure 13 and stacked column graphs which can be seen in figure 14.

3.4.3 Pie Chart

The pie chart is used to show how different data is distributed over the entire set. For example figure 13 shows which application is calling the most permissions, which is shown in a percentage value. The reason that it is shown in percentage is because when the database grows, the numbers can get really big. This abstracts great numbers that would be hard to compare to each other at a glance, while percentages can at a glance give a quick impression of the situation.

3.4.4 Stacked Column Graph

The stacked column is instead used to show behavior over a 24 hour time period. Where the x-axis is the time and y-axis is the value. It is stacked because it needs to show data from different source’s at the same time. This can be seen in figure 14 which shows how permissions are called upon over a 24 hour period. This is a perfect example to what is discussed in section 3.5. Because of the
Figure 13: Example of a pie chart presented to the user, displaying how many permissions different applications use.
Figure 14: Example of stacked column chart presented to the user, displaying how different permissions are called upon over 24 hours.
graphs tendency to become cluttered, the user can specify a time period to reduce the amount of data being showed. It is not recommended to use this graph to display too much data but instead use limits to reduce it.

3.5 Discussion and assessment of design

A big drawback with the current solution is that the user needs a rooted Android device, which the majority does not have. This means that the majority cannot utilize the application themselves to collect their own data. Because of this the majority will only be able to use the web UI with data from device’s that are not their own. This a huge drawback and will most likely make people less interested in using the web UI. But, unfortunately there is no solution right now on how to bypass the need of a rooted device.

There is also a problem with the graph showing permissions used over 24 hours. The graph is very big and is hard to fit on a mobile screen, especially when in portrait mode. The graph can not be squeezed more because that would result in data being rendered over other data. So, this application is best suited for a larger screen, especially the third graph. The bigger the browser window is the better the results will be rendered.

4 Implementation

This section gives a detailed implementation description of the KAUDroid system and its most important elements.

4.1 Android Application

Each of the different classes within the application are designed with the single responsibility in mind. The classes extend the functionality from activity, service or intentservice depending on what the class is supposed to do. The names of the classes describes what they are intended to do. The rest of this chapter will describe some of the different classes.

4.1.1 MainActivity

This class is as the name suggests a class that extends appcompactactivity which means it is connected to a UI, the home screen see figure 4. This class handles the start of the application, all requests for permissions, the start of services and the interaction with the UI (button clicks). The services starts after the user confirms the applications permission requests since these services uses those permissions.
4.1.2 SendDataService

This class extends the intentservice described in section 2.1.4. It is responsible for sending all the unsynced data to the server. The IP and port that are used to send the data to are collected from the preferencemanager before starting the background thread, the preferencemanager holds all the applications settings as key value pairs. The background thread then goes through the entire database and checks the sync bit to see if the specified entry should be sent or not.

If there is data to send the Diffie-Hellman key exchanges starts before starting the loop to send the data. The key created with Diffie-Hellman is used to encrypt and decrypt the messages between the server and application with the encryption module, see section 4.1.6. A simple protocol for the transmission of data was constructed, see figure 11.

4.1.3 LocationService

This class extends the service described in section 2.1.3. This class collects the location of the phone, GPS coordinates. In Android the GPS coordinates can be obtained in two different ways, this class uses them both. One of them is with the help of the built in GPS, this corresponds to the permission FINE_LOCATION. The other is with the use of WiFi and cellular towers. This corresponds to the permission COARSE_LOCATION. In the LocationService class there are one function for each of the locations, fine and coarse. Both of these functions have a locationlistener. This listener receives broadcast from the Android OS when a new location has been found together with a couple of parameters. These parameters include accuracy and time which is used to determine if the static GPS variable should be updated.

4.1.4 loggerService

Another class that extends the service class is described in section 2.1.3. This class is quite demanding so in order to not affect the UI a separate thread is created within the service. The created thread queries all installed applications on the device, for the time of the last use of the application permissions. This is done with the help of the application App Ops which was removed from Android 4.2 which is why a rooted phone is needed. The command Shell.SU.run("appops get " + appName) is done within the newly created thread for all installed applications and returns the time all permissions the specified application has requested. The response is then checked against the local JSON file to see that it is not present and that the time of use is newer than the static lastrun date variable. If both are true it is added and if not the permission have already been added to the JSON file. The lastrun variable is needed since synced entries is deleted from the JSON file at an interval specified by the user, see section 4.1.5 for more info.
4.1.5 deleteSyncedDataService

DeleteSyncedDataService is a class that extends IntentService as the class SendDataService. This because they both operate in a similar way, do the intended work and finish. Which is what the IntentService offers in a separate thread to not burdened the UI thread. The implementation of how we delete entries is quite straightforward. To start with a JSON array is created and filled with the entire local database. Then inside the IntentService’s worker thread there is a for loop to iterate over the entire JSON array with entries. Then a check is done against the sync bit for each JSON object within the array. If it is synced it is deleted, if not nothing is done with that entry. After iterating through the arrays, the new array is written to the database.

4.1.6 Cryptography

This class is used to conduct the encryption and decryption of messages that is sent between the server and the application. It is a standalone class to make it easy to change encryption method. It only contains four methods and a couple of static variables that for example defines: key length, encryption algorithm and the numbers used for the Diffie-Hellman key exchange (p, g). The methods in this class are encryption, decryption, generateSecureRandom and generateIV. Both the encryption and decryption method are made to be easy to modify for example other key lengths and algorithms which the static variables is used for.

4.2 Database

As mentioned in section 3.2 the database functionalities is implemented in its own module. This to not affect other parts of the project when changes are made on how access, use and updates to the database are done. This module consists of three different classes, each one fulfilling important parts to make it work as intended. The three classes are DatabaseHandler, DatabaseQueries and databaseParser. The relation between the three different classes is explained below and visualized in figure 15.

Figure 15: Database class diagram.
The databaseHandler class is the most important one. It contains methods that are essential for KAU/Droid to work correctly. It implements methods such as connecting and disconnecting to the database on the remote server and adding the information gathered from the phones to the database. Ability to do queries on the database as well as multiple different methods that generates the correct SQL-query were also implemented. These SQL-queries are needed to fetch the specific information wanted. It is a central part of the overall functionality of the database module and is the only class that is accessible from other parts of the project.

The databaseQueries class contains methods that are only able to be called by the databaseHandler. It basically generates string based SQL-queries based on what the databaseHandler is interested in. If the databaseHandler wants to fetch information about how many times a specific permission that all applications on all phones have called, it returns a SQL-query that the databaseHandler is able to use.

The databaseParser class has the job to parse different data so that it is easy to use and inter-operable with the rest of the database module. It is used to get out the relevant parts of the information sent from the mobile devices. It contains methods that, for example, grabs the GPS-coordinates from the incoming data so it may be added without difficulty to the database. Furthermore, it also involves methods that checks for the correct format of parts that are sending information between each other. This to prevent unexpected formats that may cause errors.

4.3 Server

The server is divided into three main parts, HTML Server, KAU Server and Administration tools. Each service executes as its own process. The HTML Server is responsible for communicating with the web client, it fetches user-requested data from the server and sends it to the client. The KAU Server communicates with the Android clients. With the help of a simple text basted protocol it receives information about Android application usages and stores it in the database. The last major part is the Administration tools which take command line input from the user and performs the predefined administration command on the server. Object instances of the three parts are created in the main class and then runs independently of each other.

The following three subsections present implementation details about the different services. Each of the following subsections presents a class diagram representing that part of the system. A complete class diagram can be found as additional material in appendix A.

Listing 1 visualize the implementation of the three subparts to give a better understanding of how they are used to complete the system. At server startup three objects are created admin, privacyServer and HTTPServer. The privacyServer object and the HTTPServer are placed and executed in two new threads. The main thread stops in the admin method chooseTool, where it waits for the operator to end the program. When the right input is received by the chooseTool method the program continues by interrupting and waiting for the two sub-threads to finish executing.

25
Listing 1: Implementation code explaining the KAUDroid system life cycle.

```java
AdminTools admin = new AdminTools();
privacyServerThread = new Thread(privacyServer);
httpServerThread = new Thread(HTTPServer);
System.out.println("KAUDroid starting");
privacyServerThread.start();
httpServerThread.start();
// Until stopped by user input
admin.chooseTool();
privacyServerThread.interrupt();
privacyServer.disconnect();
httpServerThread.interrupt();
waitForThreads();
System.out.println("KAUDroid stopped");
```

4.3.1 HTTP Server

As seen in figure 16 the HTTP server is started from the main part of the program in its own thread. The structure of the HTTP model is to a large extent implemented and designed by the Apache Spark framework (2.8). Three classes were created by the framework, filters, routes and the server class template used by the framework to share the server state between different threads. The filter class contains client request filters evaluated before and after requests. As for now no particular filter is implemented but an example could be authentication of requests. The routes are a major part of Spark and consists of three sub pieces starting with a verb. The verb explains the intention of the client request, this could be posting, fetching, deleting or adding new files. The second part is the path to the file that the client requests an action on. The last piece of the route is the callback, this is the code which will be executed for a particular route. In order to fetch information from the web client the routes will be depending on the DatabaseHandler.

The RouteSetup class consists of two routes. The first route fetches phone id numbers, application names and permission names from the database. This information is used by the second route to select a data set on the server to visualize. The two routes return different data sets but both returns the same HTML page used to represent the data.
4.3.2 Privacy Server

The Privacy Server handles communication with the Android clients. When the Privacy Server object is created in the main method it opens a port and waits for new clients to connect. For each new client a new IP port is opened and the communication moved to this port. This is done in order to make it possible to handle multiple clients at the same time. Communication on the new port is handled by a new instance of the communicationHandler. Each communicationHandler instance is responsible for communication with a particular Android phone. Figure 16 shows a dependency to the databaseHandler explained by the need to store information sent by the Android phones. The cryptography class is used to secure the communication between the server and the phones.

For each new communicationHandler an instance of the clientConnectionInformation is added to a list. This object contains references to the communicationHandler object and the thread it is executing in. When a client terminates the connection to the server this object is removed from the list.
As seen in section 3.3.2 information sent by the Android client is validated in order to keep information consistent in the database. This means that all information passing the validation test will be added to the database. If the information is invalid the Android client will be notified. The validation test can be seen in listing 2. To validate the input the method uses a pattern, which is matched to the method input parameter. The advantage of this method is that it requires all expected fields to pass and at the same time enables the fields to be empty. The disadvantage is that category associated information is not validated. This means that a non-existing permission could theoretically be added to the database.

Listing 2: Implementation code used to evaluate android client input.

```java
public boolean hasRightFormat(String inputDataFromApp) {
    inputDataFromApp = inputDataFromApp.replace("", "");
    inputDataFromApp = inputDataFromApp.replace("} ", "");
    inputDataFromApp = inputDataFromApp.replace("\" ", "");
    String pattern = "PhoneID: \S+, Package: \S+, Permission: \S+, Timestamp: \S+, GPS: \S+, Synced: 0";
```
if (inputDataFromApp.matches(pattern))
    return true;
else {
    System.out.println("Invalid app data");
    return false;
}

4.3.3 Admin tools

The Administration tools implements a set of possible operations which is presented in a terminal printed menu. Through the system.in method the operator is able to communicate with the program. A switch-case statement is used to execute the right command. As for now the tools is mainly used to give the operators an easy and secure way to stop the server.

4.4 Web User Interface

The web UI consists of only one view. It is written in HTML and uses Bootstrap as a design framework. This view uses information sent by the HTTP Server (4.3.1). It is sent as a HashMap and the view uses a Thymeleaf engine to extract the data. There is a second method which is used for passing information, which is a JSON object. The object is also sent in the HashMap. The data that Thymeleaf uses is for the site itself, filling in names or options in the drop-downs for example. While the JSON data is for rendering the graphs.

4.4.1 Graphs

There are three different graphs, each is constructed with JavaScript. With the help of Thymeleaf a JavaScript will only be executed when the correct data is passed. If there is no data for a JavaScript the div tag and its content plus the JavaScript tag and its content will not be included in the HTML code that is sent to the client. If the correct data is passed the JSON data will be collected from the HashMap and converted into a JSON object. This object is then used to create the graph, which is done with the framework Highcharts. Highcharts allows the graph to be built on dynamic data. So, depending on the JSON object the graph will be different. Thymeleaf is also used by the graphs to fill in the selected queries that the user have made, see the title’s in figure 13 and 14.

There are two different types of graphs, a pie chart and a stacked column graph. Both of them are very similar but have some minor differences in the JavaScript. The pie chart only have one value connected to a name. While the stacked column graph has an array with 24 different values connected to a name. The reason for this is, in the stacked column graph a name can have different values over 24 hours. The JavaScript for both graphs were created from templates
provided by Highcharts and then received some minor alterations for a perfect fit.

5 KAUDroid Project

For the official project description see appendix C.

KAUDroid was developed over a semester by four project members. On average the amount of time spent on the project each week was 20 hours. This meant that time was a hugely limited resource. Because of that, time could not be wasted and at a certain point implementing more functions was not an option. Without a doubt, there are many ways which KAUDroid could be further improved and developed, see section 7.

5.1 Development of the Project

The development of the project was calculated to take 16 weeks of work, including the report writing. Since it was a big project it would require a certain bit of planning for the optimal progress. This meant that the first task in the project was to create a project plan.

5.1.1 Start Up

It was clear from the beginning that the platform would consist of four modules. A mobile application, a server, a database and a web user interface. All of these are described in detail in section 3. It was clear how these modules would co-exist and communicate with each other.

The first problem that the project team faced was creating a time planning for the project’s lifetime. Since all of the project members had no previous experience in this it proved a challenge. But a rough time plan was created, see figure 18 which later received changes.
Figure 18: First Time planning

Figure 18 may contain some cryptic words, for more information what they mean, see below:
• **Start Up**: Reading through project description and then moving on to the architecture of the platform. This meant deciding the design of our solution, what different components and how will they communicate. Choice of what technologies that would be used for the different components.

• **Communication Handler**: The part of the server that would be able to communicate with the application and receive data from the application.

• **Web Server**: Creating a web server that could be accessed with a browser.

• **Private Communication**: Making sure that the data that is sent between server and application is encrypted.

• **Database**: Design, a handler and a parser for the database.

• **WebUI**: Start working on on the graphs and what information they would show. Also how the website would look in general.

• **Getting familiar with application**: The application for the platform already existed, but needed improvement. This required some time to understanding how the application worked and how it could be improved upon.

• **Connect application to server**: Sending data between the application and the server.

• **Admin Tools**: Tools that would allow adding and removal of data from the database while program was running. Was also planned to include more tools for the server.

• **Database to Web**: Sending data from the database to the web site.

• **Auto Sync App**: The application should automatically push data to the server on a given time.

• **Visualization**: Visualizing data from the database on the web site through graphs.

• **Permissions**: What permissions was the most interesting.

• **Solve Problems**: Polish of the code, project report and other aspects of the project.

Each member of the project had undertaken a specific role for the project, the following four roles were assigned.

• Project Manager - Christian Pedersen

• Technical Project Manager - Gustaf Söderlund

• User Interface designer - Adrian Carlsson

• Software Tester - Fredrik Persson

This would only indicate who were responsible for different areas of the project. But when it came to decisions in all areas these were 100% mutual and taken as a group.
5.1.2 Taking action

When the initial planning were complete and an idea for how the road to the end product looked the first initial steps were taken. These steps included building the server, database, web site and Android application. The framework which the whole project would build upon. At this stage it was obvious that the initial time planning would not be correct for long, this was in week 37, which was the second week of the project. Certain parts that were estimated to take a long time were completed in under a day. At the same time the first design for the database was changed for a new and improved one, this happened multiple times during the project.

The group members would dynamically move around and help each other when the time estimated was not enough or problems had arisen. Which helped keep the group on track, so this was a practice that were continued throughout the whole project.

Since the time planning was coarse there was a need for a more detailed planning as well. This was achieved by daily and weekly morning meetings. In the beginning of each week the upcoming week was planned. Then each day at the morning meeting group members reported progress and estimated times for completion. This resulted in each group member knowing what the others were doing and if there was a problem it could be solved as a group. During these meetings design was also discussed. Previous to development starting the design was chosen as a group. For some design areas, as the database or the graphs longer discussion were held and often the responsible would come with suggestions of design. Then a decision was taken as a group.

Since all group members were fully aware of this report coming up later, all decisions taken were written down. This would prove useful in the writing of the report.

5.1.3 Mid Project

As the project progressed there was a big crossroad ahead. The previous parts of the project had been the technical part, creating a database etc. Now design and visualization became relevant. As discussed in section [I] it is important that the graphs are simple and easy to understand and at the same time shows crucial data. A lot of time went into the graphs and how they would look. Many discussions in the group and even going as far as to asking friends regarding what they would want to see. At this time in the project a choice had to be made as to how the visualization would evolve. There were a lot of ideas on ways to visualize, where a lot of them had to be set aside while one design philosophy was chosen.
5.1.4 Wrap up & Report

One to two weeks before ten weeks in, a decision was made to not implement new features and instead polish what existed. This was in preparation for the report writing so that it could be the focus in the upcoming weeks. It is hard to halt the development when there still are many ideas of extensions that would be a great fit.

The report was divided up so that each project member could write regarding what they had developed and their area of responsibility.

5.2 Roles

Each member in the project had a different role and a different responsibility. This was to ensure that different aspects of the project were handled and that there was a responsible person that could ensure that progress was achieved in different areas. This made sure that parts of the development would not fall through the cracks. The roles were assigned before the projected started and would persist all throughout the projects lifetime.

5.2.1 Project Manager

The project leader is responsible for the project as a whole, making sure that the development progresses according to plan. It is the responsibility of the project leader to make sure that all project members are aware of their work tasks and making sure that the work progresses according to schedule. Communication between the customer is also handled by the project leader and since there was a report which was reviewed there was also communication with a person responsible for the reviewing. During the report writing there was a lot of working from home, here morning meetings over group calls was managed by the project leader. The project leader is also responsible for keeping the group informed and in aiding of the integration of different modules and making sure that all are on the same page.

5.2.2 Technical project manager

During the development process the role of the technical project manager has been to take implementation and design decisions regarding project architecture. I have coordinated the actual code implementation and had a superior vote on which frameworks and development environments to use. Comparing advantages and disadvantages to convince other developers why to use a specific technology or programming language. A major part of the implementation time have been dedicated to the database and the KAIDroid server.

5.2.3 User Interface designer

For the duration of the development the UI designer has been responsible for selecting graphs and visualizations methods. Furthermore, a bigger responsibility
to how these ideas should get implemented together with libraries and classes. I have had a heavier opinion on the layout of the UI, both on the application and the website. This means that I have been presenting ideas on how the data should be visualized so that users can easily understand what they see. I have been in charge of ensuring that the UI the users are interacting with is correctly communicating with the code. I have mainly worked with the communication between the web and the database which includes database queries, to fetch the interesting data, and how this data set should be visualized.

5.2.4 Software tester

The software tester is responsible for that the delivered product holds a certain quality. This is achieved with the helps of different tests in order to find bugs in the code base and make sure later changes does not create new bugs or interfere with other parts of the code. The designed tests for the product has been discussed and written with the help of the other team members. In order to test the code base as efficiently and fast as possible due to the limited time frame. As the software tester my vote of how to test certain parts of the system in case of disagreements had a higher value.

6 Testing

Testing is a vital part in any software development situation. It verifies that the software will meet its corresponding requirements in functionality, usability, reliability, security and performance. It verifies that the software that has been developed also meets the user needs. It improves the overall quality of the product by finding code defects early and is a good way to find out in which state the product is in.

The platform has been tested with both unit and manual tests based on what was considered the best and most effective in our time limited case. The platform is tested to verify that it is in a stable state and ready for use.

6.1 Manual Tests

Manual testing have been used for certain parts of the project where unit tests were either not an option or not an optimal solution. Generally, manual testing is more time consuming and could not be considered fully reliable due to human errors that may occur. The reason behind manual testing on some parts is explained in their respective sections down below.

6.1.1 MySQL Server

The design of the database tables has been tested manually. Every time the design has been changed or updated, manual tests have also been made so that the data to be entered ends up in the right tables. This was done by simply adding and fetching data using SQL-statements directly in the MySQL software.
Given that none of the project members had previous experience of testing on a database this was decided to be the efficient choice.

### 6.1.2 Database module

In the database module, much of the functionality could be tested with unit testing. The class that had some methods unavailable to test automatically was the databaseHandler. The reason for this was that most methods did not have return statements in that class. They are simply used for arranging the correct SQL-statements based on what the user are interested in and send this to the database. A solution to this would be to create a mocked database for just the testing purpose. Since none of the project members had done this before, it was decided to focus on other things instead. This to save time. These methods were instead tested in the same fashion as the MySQL server above. Since the application is quite hard to test with GUI functions the test here has mainly been manual. Additionally every function that accesses any of Androids internal APIs needs to be mocked. So for each new function added we have tried it out and looked at the log messages at logcat. Then looked at the servers log message to see if they comply with each other. Some things have been much harder to test here such as the GPS since the phones GPS is extremely bad.

### 6.1.3 Web User Interface

The Web UI was tested with manual tests. The focus of the tests were to test inputs and make sure that it would perform the right task for each input. But, also to detect faulty inputs and disregard them.

### 6.1.4 The application

Since the application is quite hard to test with GUI functions the test here has mainly been manual. Additionally every function that accesses any of Androids internal APIs needs to be mocked. So for each new function added we have tried it out and looked at the log messages in Android Studio. Then compared these with the server log messages to see if they comply with each other. Some things have been much harder to test here such as the GPS since the phones GPS is extremely bad. The manual test used for testing the application can be seen in appendix D.2.

### 6.2 Unit Tests

Unit test saves time and is more efficient then manual testing. It helps with the human error probability and generally improves testing coverage and the reliability of the platform. Different testing tools may be used for different scenarios and it helps developers find bugs and errors quicker. Down below are the parts where unit tests has been done. As well as the justification behind it.
6.2.1 Server

The unit testing for the server is carried out in as large extent as possible. Code wise the project is tested up to 40%. The reasons we could not reach a high coverage was for two reason. Our inexperience with overall testing and how to implement easily tested code, heavily affected the testing coverage. In the code there are a lot of methods that does not have a return statement, meaning that their contribution is some type of side effect. An example of these side effects could be open a new port for Internet traffic, generating a new thread or receiving a message from a client.

Another reason for the relative low code coverage is the scope of the project methods. During the development we applied the Information Hiding principle, this implies that methods inside a class that was not supposed to be used outside of that class, would be declared as private methods. The disadvantage of this was that they also were out of scope of the testing class.

7 Future work

As mentioned several times in the report, there is so much more that could be added to the KAUDroid project. In this section are a list of areas that can be further improved or even extensions that could be added to improve the KAUDroid platform.

7.1 User Interface

There is nothing advanced with the current web site and the UI of it. This is something that could use a face lift to increase the appeal of it. Some people will definitely straight up ignore this page for its simple UI. Unfortunately there was not enough time to fix the aesthetic of the page. Instead priority was put on the data shown in the graphs and having a working system.

7.2 Visualization

In the project’s current state there are 3 different graph’s for the user. There is so much more potentially interesting graphs that can be included be for the future. Graphs that would show different data than what is available now through graphs. There could also be algorithms that can find specific behaviors in data sets that are interesting where automatic graphs would be created and shown for users.

Since there is support for GPS coordinates it is possible to link permissions calls to a geographic location. A map service, for example Google Maps, could be use to highlight where permission calls are made and at what intensity. Showing if there are more permission calls at different geographic locations. It could hint towards applications being more active when surrounded by other phones for example.
7.3 Data Gathering

With KAUDroid a platform exists for collecting data and storing it. One of the next steps is to start gathering data and using the platform to examine the data. Data gathering should be done in an orderly fashion. Setting up points for how devices are to be used and compare how different behaviors differ. Through this it is possible to detect deviations from expected behavior. It can also be used for indications that some applications are behaving in a malicious manner.

It is very important that the data gathering is done thoroughly for the results to actually matter. It requires coordination. This is the major reason no specific data was gathered during the project. It would require a lot of time and focus. During the project both of these resources were focused at the development platform instead. Data was gathered during the project but there is not much meaningful information that can be extracted from it.

7.4 Battery efficiency

The application is of today very demanding. It uses the GPS frequently and queries all installed applications in the interval specified in the settings page. Both this task would be good to optimize both for performance and for battery efficiency. Due to the time limit, the focus was on getting it to work correctly and not anything else.

7.5 Lose the need for a rooted phone

As of today the application need to be rooted in order to work with the application. This because the application uses and old Android application called App Ops that is used to collect the permissions. This application was removed from Android 4.4.2 [25] and thus to install the correct framework to be able to use it, the phone needs to be rooted. This greatly limits the potential to reach the public of the application, since most lay persons do not use a rooted phone. To reach the lay persons a version of the application that could be used without rooting the phone would be needed to be developed and released. Advantageously through Google play store.

8 Problems

Apart from minor implementation and design difficulties explained in section 3 and 4 the development process has also encountered some environment dependent problems.

The tablets used to gather test information and for the application development testing did not have processing power enough to smoothly run such a demanding application as KAUDroid. The application have been tested on a smartphone with an newer Android version and better performance. In this case the Android OS killed the process after a couple of hours. The reason for this is not
established, but a strong theory is that the OS has a lack of memory and then kills low priority applications. Without access to the internal OS there were no solution for this problem. As for now the application can be restarted to enable the background processes again.

Another difficulty was that the server hardware once a day restarted for maintenance purposes. The problem of an unreachable server is covered in the implementation which means that no errors appeared. But as a server operator the process of restarting will get irritating when done multiple times, even if just a few clicks are required.

At the start of the project it was possible to test code implementation and functionality locally. But when introducing the remote server, testing of new changes became much more difficult. For server development the IDE Eclipse was used, both for code writing and to compile the Java project. To remotely execute the code on the server the IDE was used to export an Executable JAR file which was transferred to the server via FTP traffic. This was an inconvenient development procedure and was soon replaced. The Maven build tool was installed on the server to compile the project and Git was used to transfer the code.

Throughout the project the design of the database structure has been a recurrent problem. Since there was no way to verify that a given structure was better than another, many different designs have been tested. Aiming for as little redundant data as possible, table-structures and sizes have varied during the development process.

If there is any implementation problem that should be highlighted it is the difficulty introduced when allowing both server and Android client to close the connection. When an Android client closes the connection the listening method will cast an exception which is caught in a controlled way. When the server wants to close all the connection it needs access to the close method in all currently running threads. This problem was fixed by introducing the client-ConnectionInformation class and saving all created references.

9 Acknowledgement

We would like to express our gratitude to the JavaScript framework Highcharts for being allowed to use their framework under a non-commercial license. This framework is used to create all the graphs for the WebUI. Figure 13 and 14 are rendered and exported with the Highcharts framework.

We would also like to thank Lothar Fritsch (Associante Professor In Information Security) and Nurul Momen (PHD Student), both from Karlstad University. To thank them for the help and guidance they have provided throughout the project. Advice that has helped us optimize the development of this project. Thank you!

3https://www.highcharts.com/
References


B  Web UI

In this appendix the web UI is documented.
Before you get started there are going to be a number of combinations that you can view data in. The data will be presented in with the help of graphs. If you feel insecure in what you are seeing or selecting what you want to see, hit the button help and we hope that it will provide you with the information you need!
<table>
<thead>
<tr>
<th>Permissions</th>
<th>All</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>All</td>
<td>Specific</td>
</tr>
<tr>
<td>Contacts</td>
<td>All</td>
<td>Specific</td>
</tr>
<tr>
<td>Camera</td>
<td>All</td>
<td>Specific</td>
</tr>
<tr>
<td>Photos</td>
<td>All</td>
<td>Specific</td>
</tr>
<tr>
<td>Video</td>
<td>All</td>
<td>Specific</td>
</tr>
<tr>
<td>Twitter</td>
<td>All</td>
<td>Specific</td>
</tr>
<tr>
<td>Facebook</td>
<td>All</td>
<td>Specific</td>
</tr>
<tr>
<td>Call History</td>
<td>All</td>
<td>Specific</td>
</tr>
</tbody>
</table>

This information is based on the app's Settings menu and may vary depending on the specific version of the app. Always check the app's permissions for the version you are using.
KAUDroid

Before you get started there are going to be a number of combinations that you can view data in. The data will be presented with the help of graphs. If you feel insecure in what you are seeing or selecting what you want to see, hit the button help and we hope that it will provide you with the information you need.
C Project Description

Below is the project description that was the order for this project. There are some points that might not have been fulfilled, in those cases a discussion with the two responsible for the order has been held. Then a mutual agreement could have been made that some points be dropped for other features that were better fitting.
KAUdroid: App Monitoring Tool

What happens when we grant access to sensitive resources available for the installed apps on mobile phones? Numerous apps and services follow our on-line footprints round the clock, like a pack of hungry wolves! It is about time that we return the favor. This project implements an app surveillance and control tool that visualizes app access to sensitive information. A consumer protection NGO has requested a monitoring solution from you that will show consumers how much apps are extracting sensitive data from their phones.

You are the project team chosen to deliver a solution.

Motivation

In Android, the most widely used mobile operating system, an app gets access to user data through an access control mechanism called permissions. Users are asked for their consent during runtime (Android 6.0 Marshmallow and later versions) or, during installation (previous versions), in order to proceed with the app. Users are also expected to understand the consequences and make informed decisions, which is in fact very unlikely to be right. In principle, the apps are supposed to ask for permissions which are required to carry out their functionalities. Based on an initial literature survey and our primary study conducted here at Karlstad University, many of the apps are over-privileged and they have access to sensitive user data which is not required to deliver the services.

Project Description

The goal of this project is to develop transparency enhancing tools (an app and a data processing server) to visualize resource usage statistics of Android apps. As an enhanced feature, the tool shall create warnings when a certain configurable threshold of permission use is reached.

1. Logging Service: Initial investigation and background study are required to define the parameters for monitoring / listening tool. Based on your findings, design and develop an Android App that is able to -

- run as a service.
- document every resource access event (and/or other information) by the installed apps.
- store logs securely in a remote server.

This part has already been completed to some extent. You may proceed to use the AWARE prototype provided by Nurul for your project. You are also welcome to improve current solution / replace some parts / propose another one. In that case, documentation on design decision is required.

2. Server Program: write a server program (preferably in Java) capable of communicating with the app securely. The server should be able to register participating devices/profiles. Data accumulating and analyzing procedure should possess, but would not be limited to the following properties:

- Store the log locally on the mobile device in a file (preferred format - JSON) and send it to the secure server periodically (once a day) and/or manually to reduce communication frequency.
- Alternatively, send the log spontaneously and securely to the server and store them according to device ID / profile.
- Introduce ‘big data’ handling capabilities to the server by including a database and an import/export API to it. Project design documentation should include database schema and explanation.
- The server will also be responsible for analyzing the data and providing feedback for the UI (elaborated below).
3. **UI:** design and develop UIs for both web application and mobile app. The server should offer an interface as well as an API for comprehensive analysis of the collected data; i.e. overall permission usage summary for all the devices / profiles / apps. Design decisions are expected to be explained prior to the implementation phase. Some expected features are listed below:

- **Primary feature:** Visualize resource utilization summary which should include a list of installed apps sorted according to number of accessed permissions.

- **Secondary feature:** Visualize detail permission usage by each app which should include a list of corresponding permissions sorted according to number of access count. A time-line based option is also required to view usage summary of last hour / day / week.

- **Tertiary feature (bonus):** Be creative and implement an interesting visualization for the thresholds and threshold violations that can be shown on the mobile device.

**Project Management**

The project team will have four additional roles (besides implementing and report-writing) that need to be fulfilled by the project participants:

- **Project manager:** Has responsibility for planning of time, prioritization, information exchange and customer contact. Will schedule research/development/reporting work. Will collect/clarify requirements with the customer.

- **Technical project manager:** Will have decision responsibility for technical choices, selection of infrastructure, platforms, design considerations and implementation strategy. Coordinates the actual implementation work and the necessary development resources.

- **Software tester:** Designs testing criteria for quality of product (bug removal, stress testing, specific important features of the game). Development shall consider the tests to be performed. Collaborates with all other roles to design tests.

- **User Interface designer:** Will be responsible for selecting graph and visualization methods, libraries and classes. Has the responsibility to decide how visualization ideas will get implemented, both on technical and esthetical level.

**Reporting**

Each project team will have weekly meetings with their customer/project sponsor (Lothar Fritsch). At these meetings, the project leader shall report:

- Plan compliance and explain deviations from plan;

- Problems – both technical and administrative.
At the end of the project period, the team shall deliver a project report containing chapters on design, design choices with justification, testing infrastructure and test activities, security and privacy concept, and as an appendix program code in an archive.

**Expected deliverables**

The deliverables will be chapters of the final report. Intermediate reporting will best be done such that the deliverables will fit with minor changes into your final report.

1. Make a project plan with time planning for all milestones, testing, documentation and report writing.

2. Make a concept for monitoring app permission use on android devices. Deliver an analysis of techniques and methods, and select your solution. Describe the strengths and weaknesses of your solution!

3. Design an architecture for the overall system. Document your design choices (high-level design, choice of components and technologies).

4. Develop your component for analysis and visualization of the collected data! Present the chosen visualization and explain the necessary analysis algorithms.

5. Design a configuration mechanism for privacy thresholds per permission. Suggest visualizations for status overview of the thresholds and for alarms when thresholds are reached.

You will demonstrate your application and visualization during development. The project report and the project presentations are expected to show design diagrams, screenshots and sequence diagrams of the designed software.

**Requirements**

Any program code used shall be programmed by the project team. You can use open-source software and source code from the web, however you should document the components you re-use, and you should document their licensing conditions in your documentation.

If you use graphical elements, clip art, images or game character art, you should make sure that they are free of licensing obligations that will hinder re-using them in later projects. You are strongly recommended to use royalty-free imagery or Creative-Commons images with a non-viral license.

The test log data collected during development from various mobile devices will be deleted after course examination. Note however that we may import test data provided by us into the database to test features of the system.
## D Test Tasks

### D.1 Web User Interface

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Expected outcome</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load <code>/home</code> site</td>
<td><code>home.html</code> file should be loaded to browser and rendered, no graphs or their respective div &amp; script should be visible in the html file</td>
<td>As expected</td>
</tr>
<tr>
<td>Click show data with no input</td>
<td>The browser should prompt for input on <code>phoneId</code> which is required attribute. Graph scripts &amp; divs should be hidden</td>
<td>As expected</td>
</tr>
<tr>
<td>Click show data with no input,</td>
<td><code>home.html</code> should load with no graphs scripts or div shown.</td>
<td>As expected</td>
</tr>
<tr>
<td>with required attribute on</td>
<td>直接续写。</td>
<td></td>
</tr>
<tr>
<td><code>phoneId</code> removed</td>
<td>直接续写。</td>
<td></td>
</tr>
<tr>
<td>Select a phone &amp; click <code>Show Data</code></td>
<td>A pie graph should appear with data for selected device</td>
<td>As expected</td>
</tr>
<tr>
<td>Select a phone and and app &amp; click</td>
<td>A pie graph should appear with data for selected app, no device graph should appear</td>
<td>As expected</td>
</tr>
<tr>
<td><code>Show Data</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select a phone, an app and a</td>
<td>A stacked column graph should appear with data for selected permission</td>
<td>As expected</td>
</tr>
<tr>
<td>permission &amp; click <code>Show Data</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select a phone &amp; a permission &amp;</td>
<td>A pie graph should appear with data for the device</td>
<td>As expected</td>
</tr>
<tr>
<td><code>Show Data</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove attribute required for</td>
<td>For all cases the <code>home.html</code> page should load with graph scripts and divs being hidden</td>
<td>As expected</td>
</tr>
<tr>
<td><code>phoneId</code>. Then Show data for</td>
<td></td>
<td></td>
</tr>
<tr>
<td>different combinations of apps &amp;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>permissions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select device, app &amp; permission</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Within a specific time interval</td>
<td>Data will be shown from that time interval</td>
<td>As expected</td>
</tr>
</tbody>
</table>
2. With corrupt dates given (change type and enter text) Data will be shown for permission but time interval will be ignored As expected

3. With no from date but a valid to date Data will be shown for permission but time interval will be ignored As expected

4. With no to date but a valid from date Data will be shown for permission but time interval will be ignored As expected

5. With no to date and no from date Data will be shown for permission but time interval will be ignored As expected

Click Help button Help information should be displayed to the user. If it’s already displayed it should instead hide it As expected

D.2 Manual tests for the application

D.2.1 Manual Tests for main page

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Expected outcome</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click statistics</td>
<td>Browser opens on ip and port specified in settings.</td>
<td>As expected.</td>
</tr>
<tr>
<td>Click settings</td>
<td>Settingspage opens</td>
<td>As expected.</td>
</tr>
<tr>
<td>Accept all permission in popup dialog</td>
<td>All services gets started, check logcat and toastmessages to confirm</td>
<td>As expected.</td>
</tr>
</tbody>
</table>

Table 3: Main page tests

(Only those we added tested)
### D.2.2 Manual Tests for settings page

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Expected outcome</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click your phone id</td>
<td>The phones id is shown, check webpage to see that the phone id is present and the apps in the graph is present in phone</td>
<td>As expected.</td>
</tr>
<tr>
<td>Change sync time interval</td>
<td>Summary changes, time repeat on sendData is changed, confirm by watching logcat messages of sendDataService</td>
<td>As expected.</td>
</tr>
<tr>
<td>Change clear db interval</td>
<td>Summary changes, time repeat on synced data is changed, confirm by watching logcat. Check data entries are removed in logcat</td>
<td>As expected.</td>
</tr>
<tr>
<td>Change ip</td>
<td>Summary changes, check logcat to confirm we try to connect to the new ip adress.</td>
<td>As expected.</td>
</tr>
<tr>
<td>Change port</td>
<td>Summary changes, check logcat to confirm we try to connect to the new port.</td>
<td>As expected.</td>
</tr>
<tr>
<td>Check/uncheck allow use of mobile data</td>
<td>Outcome: check logcat to see that it don’t sync if not checked.</td>
<td>As expected.</td>
</tr>
</tbody>
</table>
### D.2.4 Manual Tests for locationService

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Expected outcome</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Received broadcast from os</td>
<td>The service checks the new GPS fix to compare with the old, if better update, else discard. Check logcat messages to confirm and watch server log to see permissions with GPS has the same coordinates</td>
<td>As expected.</td>
</tr>
</tbody>
</table>

Table 6: LocationService test

### D.2.5 Manual Tests for ClearSyncedDataService

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Expected outcome</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wake up alarm triggered</td>
<td>The service checks the new GPS fix to compare with the old, if better update, else discard. Check logcat messages to confirm and watch server log to see permissions with GPS has the same coordinates</td>
<td>As expected.</td>
</tr>
</tbody>
</table>

Table 7: ClearSyncedDataService test
Our mobile phone is today our most precious and confidential part of us. In many cases we do not only share phone number with it, but also our most private information. This information is to some extension naively shared and gathered without our knowledge. As the number of advanced and sophisticated services increase the more sensitive information we seem to throw at them. To prevent privacy intrusions Google's operation system Android uses something called permissions. Permissions are a decision to allow an Android application access to a resource on a specific device. Once a permission is granted there is no possibility for a user to verify that the application does not take advantage of the given trust. With granted permissions applications can collect a vast amount of information without any connection to the service it provides. To highlight this problem we present the KAUDroid platform, a two part, Android application surveillance system. KAUDroid consists of an Android application that collects permission usage on phones and a central server responsible for data storage. Information is presented to the public through a web user interface using graphs to visualize data in a understandable manner. With this tool we hope to raise the general awareness of how third-party applications tend to abuse their trust and to help people recapture their unconsciously lost privacy.